*Горшков Алексей Олегович*

*Группа 4212*

**Отчет о выполнении лабораторной работы №5 «Наложение текстуры»**

1. **Задание:** 
   1. *Осуществите рисование объекта, созданного в предыдущей лабораторной работе, при помощи дисплейного списка.*
   2. *Наложите на объект три различные текстуры на три различные грани. (Для этого необходимо предварительно создать текстурные изображения в формате \*.bmp.).*
   3. *Осуществите изменение параметров отображения текстуры объекта таким образом, чтобы при нажатии на левую кнопку мыши объект излучал свет согласно заданной текстуре, при нажатии на правую кнопку мыши объект закрашивался текстурой, и при этом моделировались коэффициенты отражения.*
2. **Листинг программы:**

#include <iostream>

#include <gl\glut.h>

#include <atlbase.h>

#include <atlconv.h>

#include <string>

#include <cmath>

#include <math.h>

#include <windows.h>

#include <stdlib.h>

#include <C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.35.32215\\include\\GL\\glaux.h>

#pragma comment(lib, "C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\VC\\Tools\\MSVC\\14.35.32215\\lib\\x64\\GLAUX.lib")

#pragma comment(lib, "legacy\_stdio\_definitions.lib")

//Переменные для вращения объекта/камеры на определенное число градусов.

GLfloat angle\_object{}, angle\_light{};

//Переменные для начала вращения объекта/камеры.

GLboolean object\_rotate{ false }, light\_rotate{ false };

GLuint list{};

void figure()

{

AUX\_RGBImageRec\* image\_1{ auxDIBImageLoad(L"texture\_1.bmp") };

AUX\_RGBImageRec\* image\_2{ auxDIBImageLoad(L"texture\_2.bmp") };

AUX\_RGBImageRec\* image\_3{ auxDIBImageLoad(L"texture\_3.bmp") };

GLuint textures[3]{};

GLuint texture\_1{};

GLuint texture\_2{ 1 };

GLuint texture\_3{ 2 };

glGenTextures(3, textures);

glBindTexture(GL\_TEXTURE\_2D, texture\_1);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image\_1->sizeX, image\_1->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image\_1->data);

glBindTexture(GL\_TEXTURE\_2D, texture\_2);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image\_2->sizeX, image\_2->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image\_2->data);

glBindTexture(GL\_TEXTURE\_2D, texture\_3);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image\_3->sizeX, image\_3->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image\_3->data);

list = glGenLists(1);

glNewList(list, GL\_COMPILE);

GLfloat MyAmbient[] = { 0.19125, 0.0735, 0.0225, 1 };

GLfloat MyDiffuse[] = { 0.7038, 0.27048, 0.0828, 1 };

GLfloat MySpecular[] = { 0.256777, 0.137622, 0.086014, 1 };

GLfloat MyShininess[] = { 12.8 };

glMaterialfv(GL\_FRONT, GL\_AMBIENT, MyAmbient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, MyDiffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, MySpecular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, MySpecular);

glBindTexture(GL\_TEXTURE\_2D, texture\_3);

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glTexCoord2f(0, 0); glVertex3f(0, 0, 0);

glTexCoord2f(0, 1); glVertex3f(4.73, 0, -4.2);

glTexCoord2f(1, 1); glVertex3f(6, 0, 2);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, texture\_3);

glBegin(GL\_POLYGON);

glNormal3f(-0.624722, 0.335006, -0.705332);

glTexCoord2f(0, 0); glVertex3f(0, 0, 0);

glTexCoord2f(1, 0); glVertex3f(1.99, 2.87, -0.41);

glTexCoord2f(1, 1); glVertex3f(4.09, 2.87, -2.27);

glTexCoord2f(1, 0); glVertex3f(4.73, 0, -4.2);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, texture\_1);

glBegin(GL\_POLYGON);

glNormal3f(0.924097, 0.332616, -0.18818);

glTexCoord2f(0, 0); glVertex3f(4.73, 0, -4.2);

glTexCoord2f(0, 1); glVertex3f(4.09, 2.87, -2.27);

glTexCoord2f(1, 1); glVertex3f(4.65, 2.87, 0.48);

glTexCoord2f(1, 0); glVertex3f(6, 0, 2);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, texture\_2);

glBegin(GL\_POLYGON);

glNormal3f(0.299203, 0.332868, 0.894246);

glTexCoord2f(1, 1); glVertex3f(6, 0, 2);

glTexCoord2f(1, 0); glVertex3f(4.65, 2.87, 0.48);

glTexCoord2f(0, 0); glVertex3f(1.99, 2.87, -0.41);

glTexCoord2f(0, 1); glVertex3f(0, 0, 0);

glEnd();

glBindTexture(GL\_TEXTURE\_2D, texture\_3);

glBegin(GL\_POLYGON);

glNormal3f(0, 1, 0);

glTexCoord2f(0, 0); glVertex3f(1.99, 2.87, -0.41);

glTexCoord2f(0, 1); glVertex3f(4.65, 2.87, 0.48);

glTexCoord2f(1, 1); glVertex3f(4.09, 2.87, -2.27);

glEnd();

glEnable(GL\_NORMALIZE);

glEndList();

}

void init(void)

{

glClearColor(0.016, 0.282, 0.259, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 50);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

figure();

}

//Функция для "отлова" событий клавиатуры (клавиатурных событий).

void KeyboardEvents(unsigned char key, int x, int y)

{

if (key == static\_cast<unsigned char>('o'))

{

object\_rotate = true;

light\_rotate = false;

}

else if (key == static\_cast<unsigned char>('l'))

{

light\_rotate = true;

object\_rotate = false;

}

else if (key == static\_cast<unsigned char>('f'))

{

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == static\_cast<unsigned char>('b'))

{

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

void MouseEvents(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP)

{

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_REPLACE);

}

if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_UP)

{

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

}

}

void myDisplay()

{

glPushMatrix();

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_DEPTH\_TEST);//Включаем тестовую глубину для реализации изображения.

//Задаем положение камеры.

gluLookAt(10, 12, 15, 0, 0, 0, 0, 1, 0);

glPushMatrix();

//Задание освещения.

glRotatef(angle\_light, 0.0, 1.0, 0.0);

GLfloat myLightPosition[] = { 8.0, 4.0, 5.0, 1.0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition);

//Включаем источник света.

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPopMatrix();

glPushMatrix();

glEnable(GL\_TEXTURE\_2D);//Включаем наложение двумерных текстур.

glRotatef(angle\_object, 0.0, 1.0, 0.0);

glCallList(list);

glPopMatrix();

glDisable(GL\_LIGHTING);//Выключаем освещение для рисования линий и прочего в цвете.

glDisable(GL\_TEXTURE\_2D);//Выключаем наложение двумерных текстур.

//Рисование координатных осей.

glLineWidth(2);

glBegin(GL\_LINES);

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(10, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 10, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 10);

glEnd();

//Рисование источника освещения

//Расположение источника света для наглядности.

glRotatef(angle\_light, 0.0, 1.0, 0.0);

glPointSize(20.0);

glBegin(GL\_POINTS);

glColor3f(1.0, 1.0, 1.0);

glVertex3f(8.0, 4.0, 5.0);

glEnd();

//Рисование направления к источнику света.

glLineWidth(1);

glBegin(GL\_LINES);

glColor3f(1, 1, 1); glVertex3f(0.0, 0.0, 0.0); glVertex3f(8.0, 4.0, 5.0);

glEnd();

glPopMatrix();

glutSwapBuffers();

}

void myReshape(int width, int height)

{

if (width / height < 1) glViewport(0, 0, width, width);

else glViewport(0, 0, height, height);

}

void myIdle()

{

if (light\_rotate)

{

angle\_light += 0.03;

if (angle\_light >= 360) angle\_light = 0;

}

else if (object\_rotate)

{

angle\_object += 0.03;

if (angle\_object >= 360) angle\_object = 0;

}

glutPostRedisplay();

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowSize(1000, 1000);

glutInitWindowPosition(0, 0);

glutCreateWindow("OpenGl\_Lab5");

glutDisplayFunc(myDisplay);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

glutKeyboardFunc(KeyboardEvents);

glutMouseFunc(MouseEvents);

init();

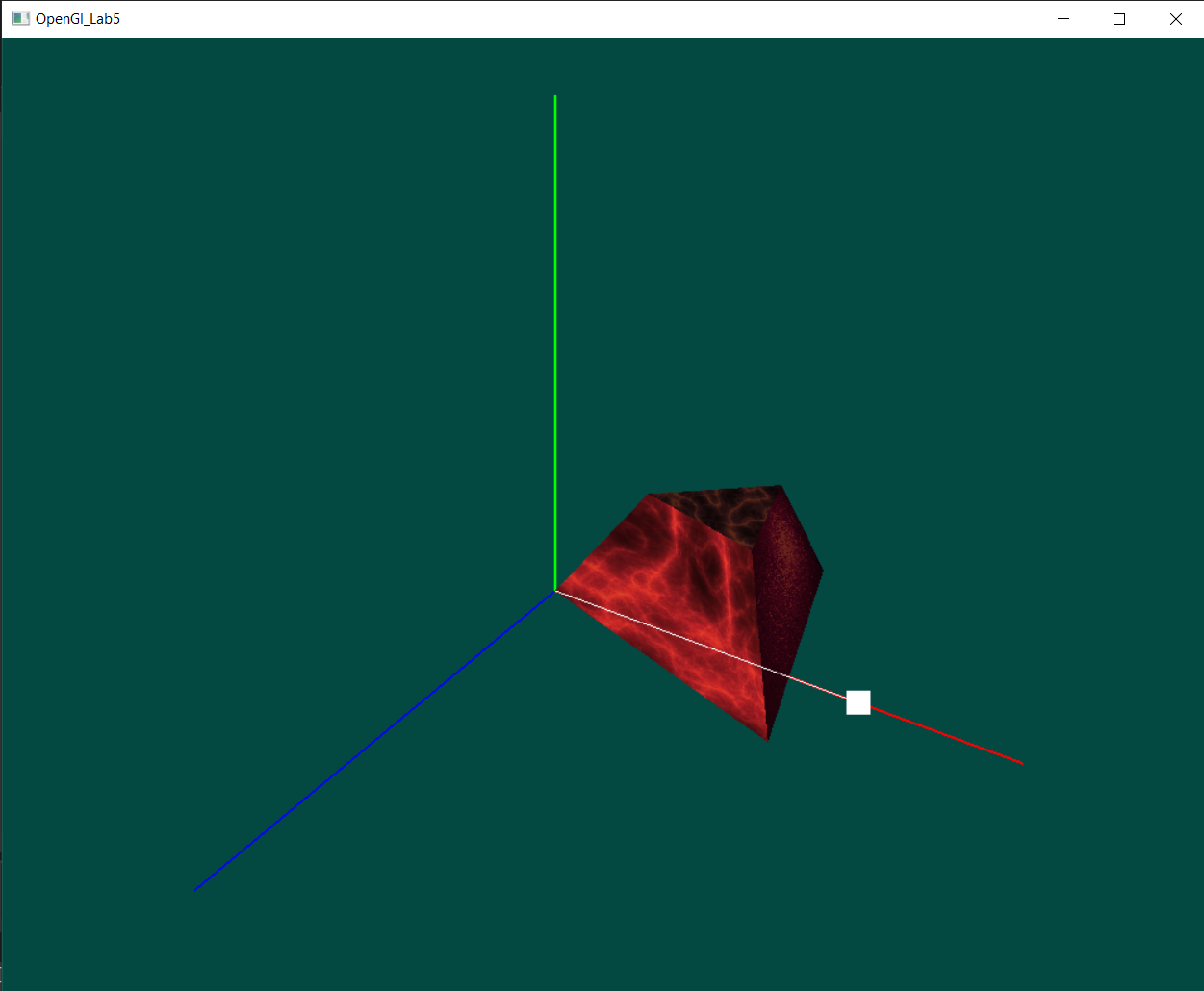
glutMainLoop();

return 0;

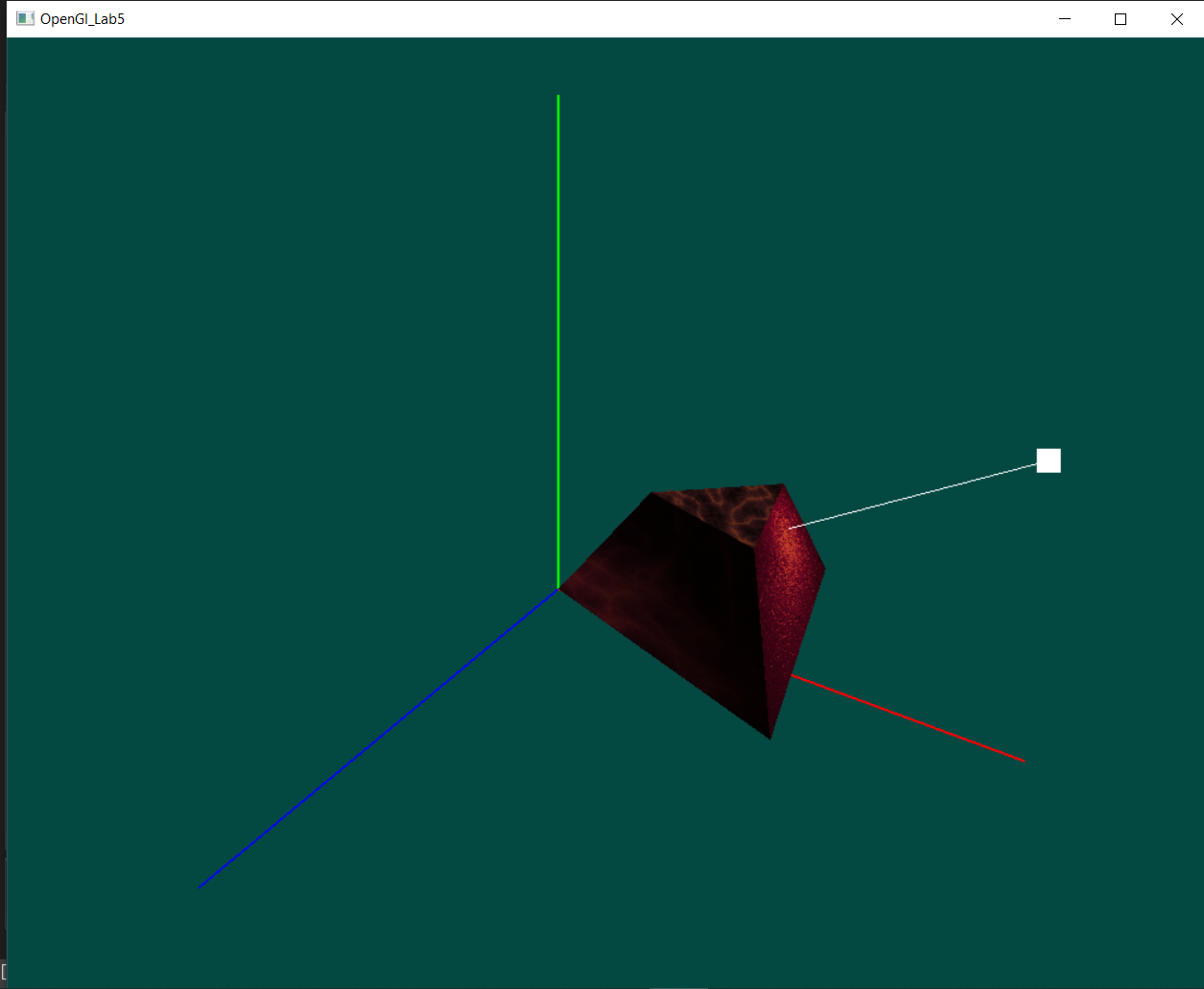
}

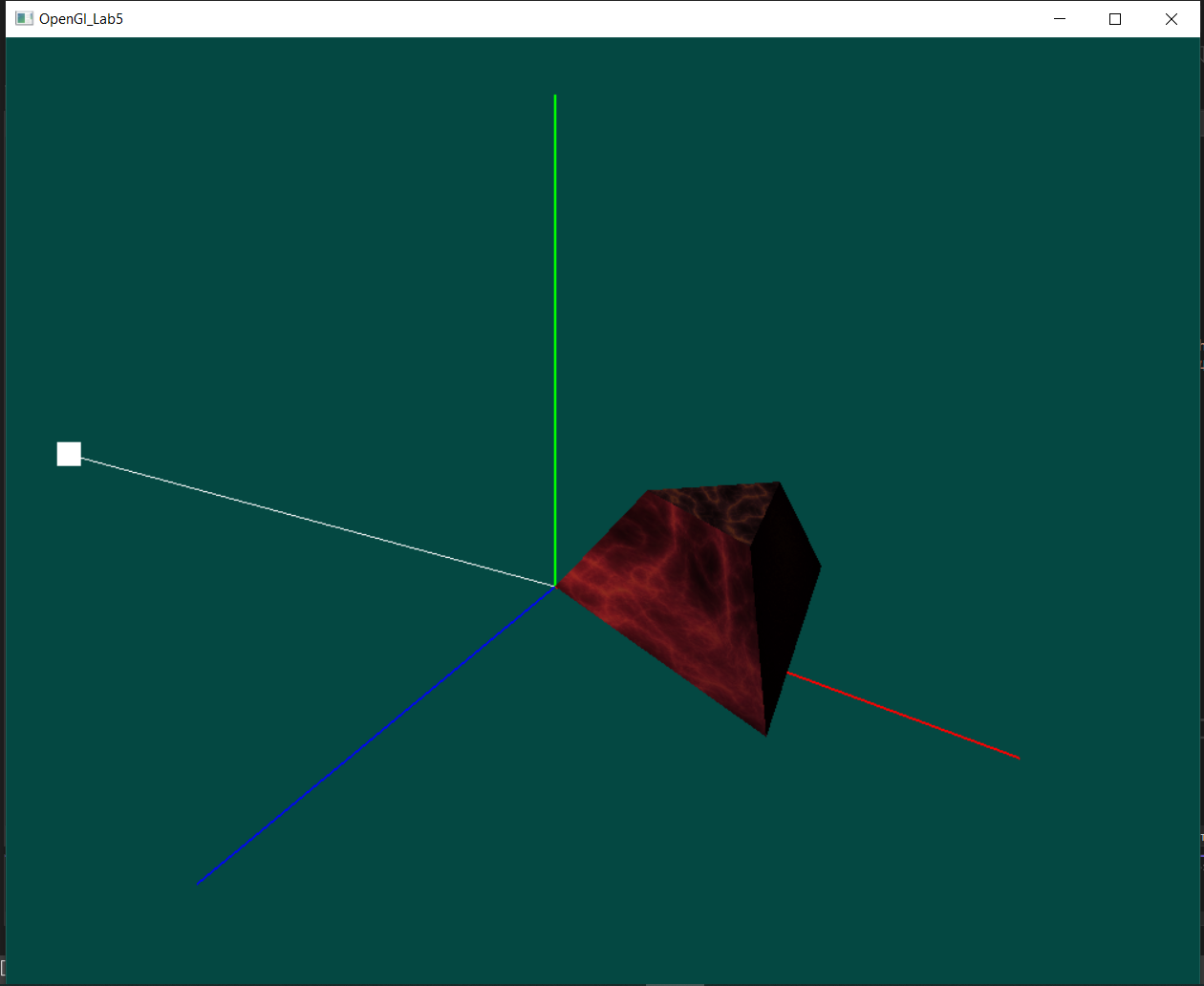
1. **Работа программы:**

*Запуск программы:*

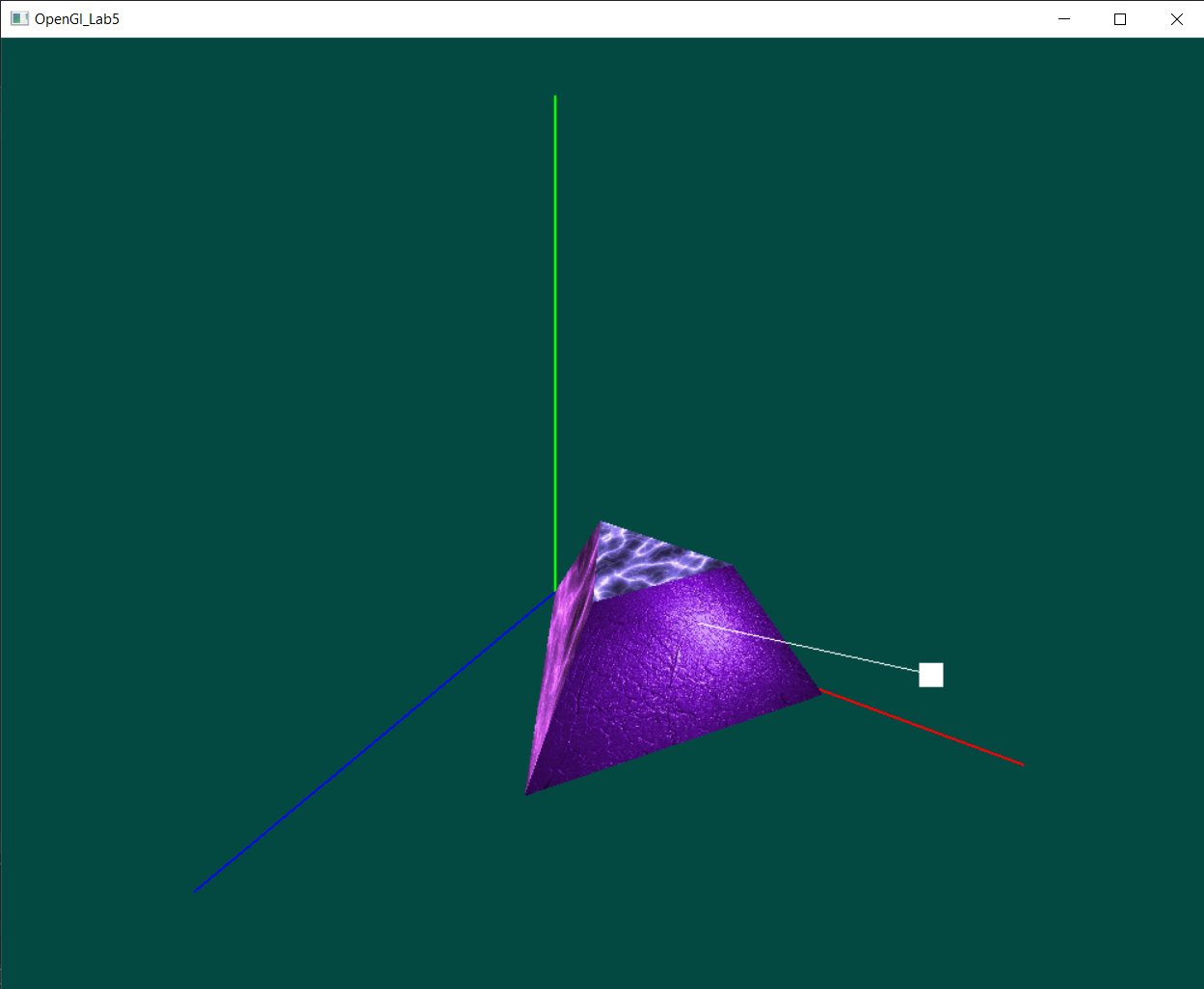


*Вращение источника света:*





*Отображение лицевых граней и создание эффекта светящегося объекта:*



*Отображение нелицевых граней с модулированием коэффициента отражения:*

